ESEC / FSE

Scalability-First Pointer Analysis with Self-Tuning Context Sensitivity

Yue Li, Tian Tan, Anders Møller and Yannis Smaragdakis

AARHUS UNIVERSITY

HELLENIC REPUBLIC
National and Kapodistrian University of Athens

Lake Buena Vista

Pointer Analysis

- Concept

  Which objects a variable may point to?

- Importance

  Fundamental for virtually all static analyses

  e.g., call graphs, alias, etc.

  Useful for many software engineering tasks

  e.g., bug detection, security analysis, program understanding, etc.

Problem: *Unpredictable Scalability*

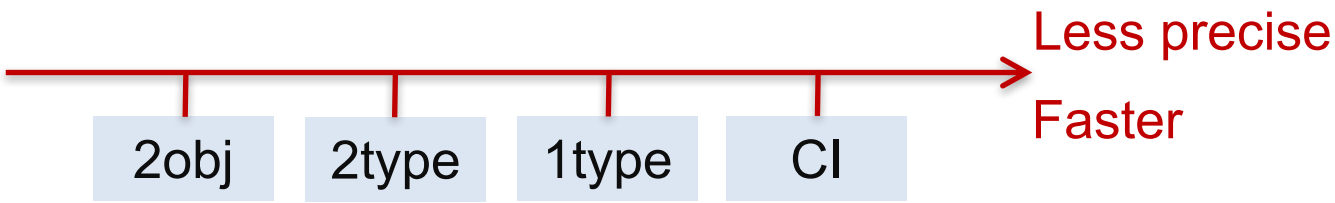
Problem: *Unpredictable Scalability*

- Precise pointer analysis is hard to scale
  - Context Sensitivity (CS): precise but slow
  - Context Insensitivity (CI): imprecise but fast

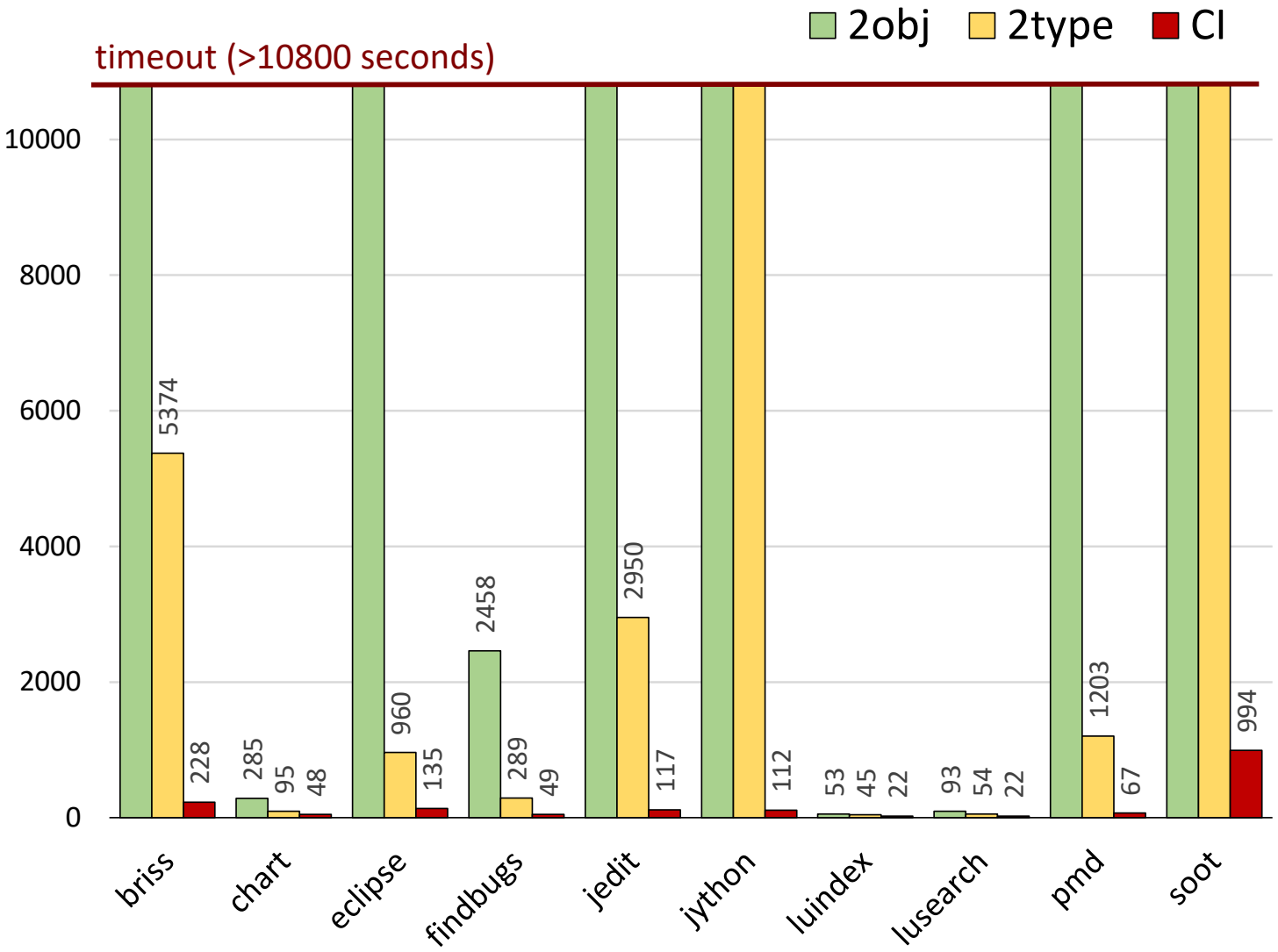
Problem: *Unpredictable Scalability*

- Precise pointer analysis is hard to scale
  - Context Sensitivity (CS): precise but slow
  - Context Insensitivity (CI): imprecise but fast

- Variants of Context Sensitivity
  - Object Sensitivity (obj)
  - Type Sensitivity (type)



Problem: *Unpredictable Scalability*



Problem: *Unpredictable Scalability*

- Scenario

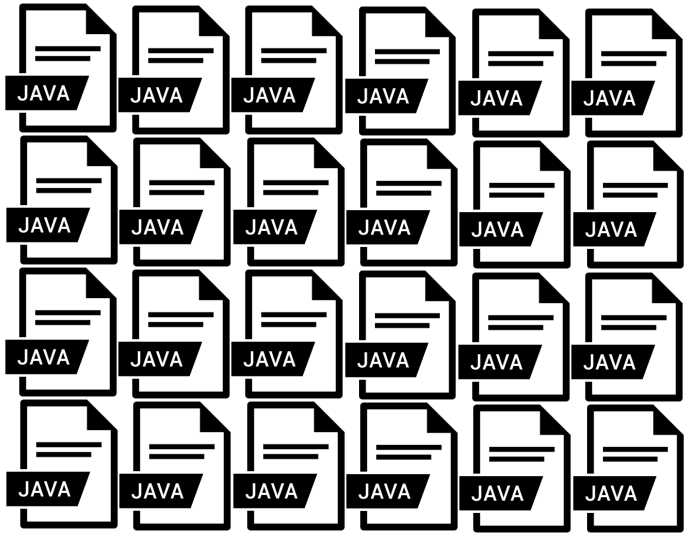
Problem: *Unpredictable Scalability*

- Scenario

as a part of a large-scale security analysis

Problem: *Unpredictable Scalability*

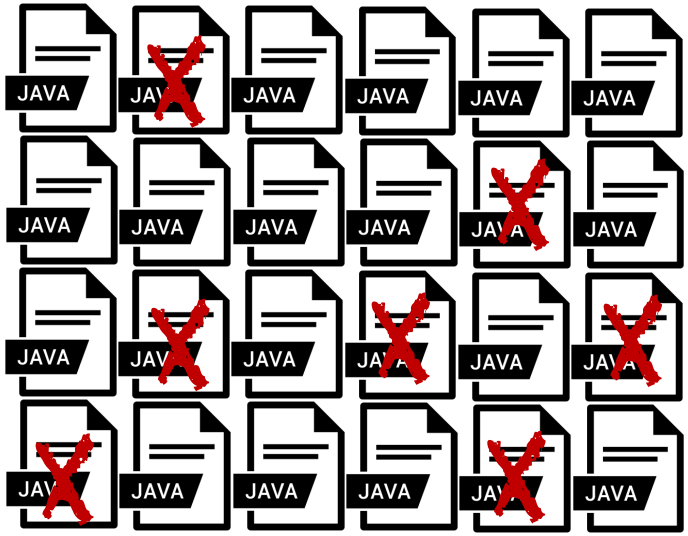
- Scenario



as a part of a large-scale security analysis

Problem: *Unpredictable Scalability*

- Scenario



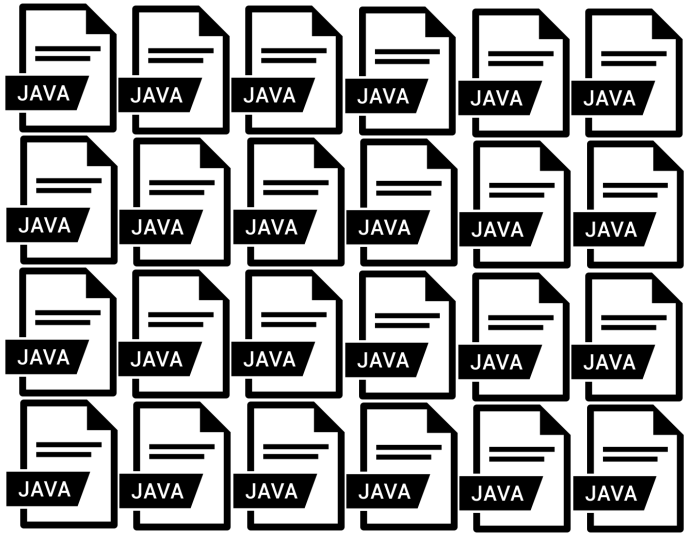
Precise 2obj ?

Unscalable for many

as a part of a large-scale security analysis

Problem: *Unpredictable Scalability*

- Scenario



Precise 2obj ?

Unscalable for many

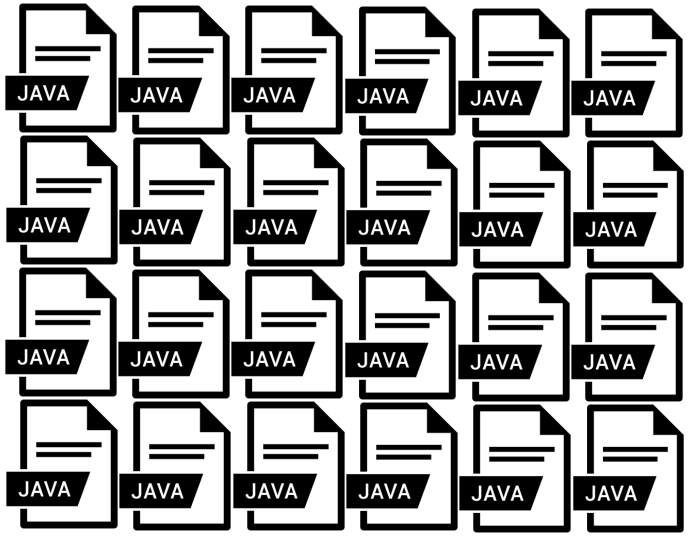
Scalable CI ?

Imprecise for all

as a part of a large-scale security analysis

Problem: *Unpredictable Scalability*

- Scenario



Precise 2obj ?

Unscalable for many

Scalable CI ?

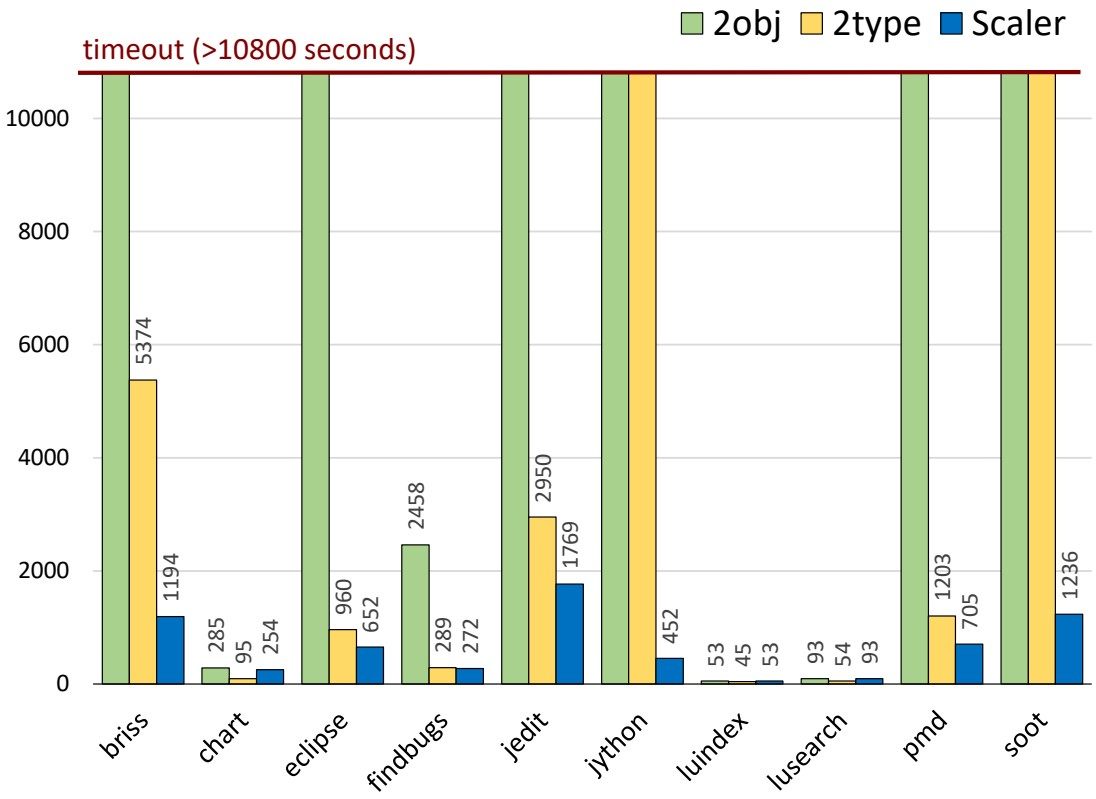
Imprecise for all

as a part of a large-scale security analysis

- Iterate until most precise that scales:
  ~~2obj → 2type → 1type~~ → CI
- Sleepless nights and still not great precision!

Good Scalability & High Precision
regardless of the program being analyzed

Good Scalability & High Precision
regardless of the program being analyzed



Scalability
as good as CI

Precision
comparable to or *better than* the best scalable CS

Idea

Key Concept

Number of *worst-case* CS points-to facts for method m

$\#ctx_m^c * \#pts_m$

- $\#ctx_m^c$: number of contexts for method m under CS c
- $\#pts_m$: number of points-to facts for method m

Insight

- Too many CS points-to facts generated for certain methods

Insight

- Too many CS points-to facts generated for certain methods
- m is scalability-critical method under CS c

$\#ctx_m^c * \#pts_m$ > ST (Scalability Threshold)

(c is expensive)

Insight

- Too many CS points-to facts generated for certain methods
- m is scalability-critical method under CS c

  $\#ctx_m^c * \#pts_m$ > ST (Scalability Threshold)

  (c is expensive)

- Identify scalability-critical method m

  $\#ctx_m^{c'} * \#pts_m$ ≤ ST (Scalability Threshold)

  (choose cheap c')

Insight

- Too many CS points-to facts generated for certain methods
- m is scalability-critical method under CS c

$\#ctx_m^c * \#pts_m$ > ST (Scalability Threshold)

(c is expensive)

- Identify scalability-critical method m

$\#ctx_m^{c'} * \#pts_m$ ≤ ST (Scalability Threshold)

(choose cheap c')

How to identify scalability-critical methods?

⇩

How to estimate $\#ctx_m^c * \#pts_m$?

How to estimate $\#ctx_m^c * \#pts_m$?

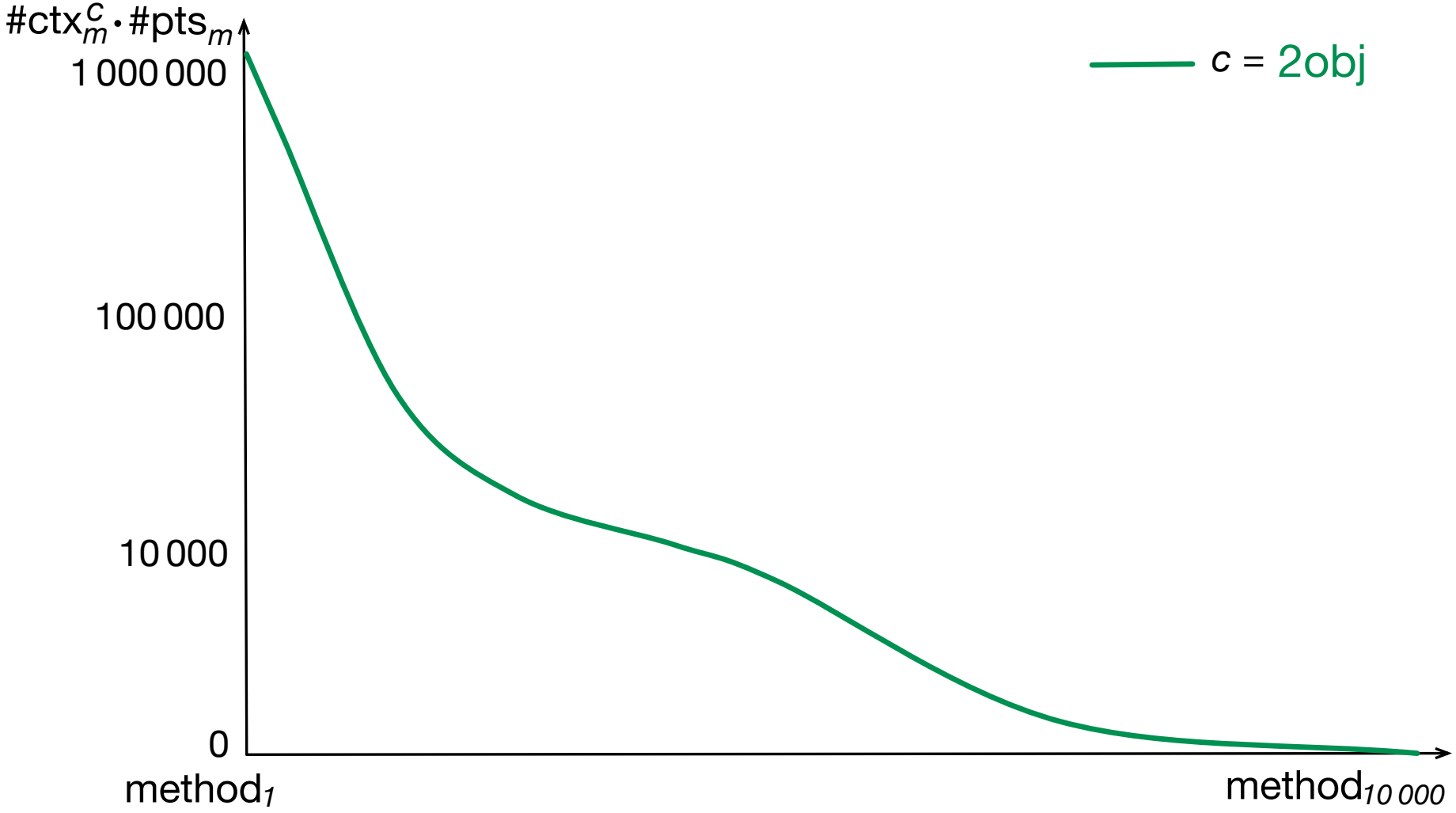
Pre-analysis: points-to results of CI

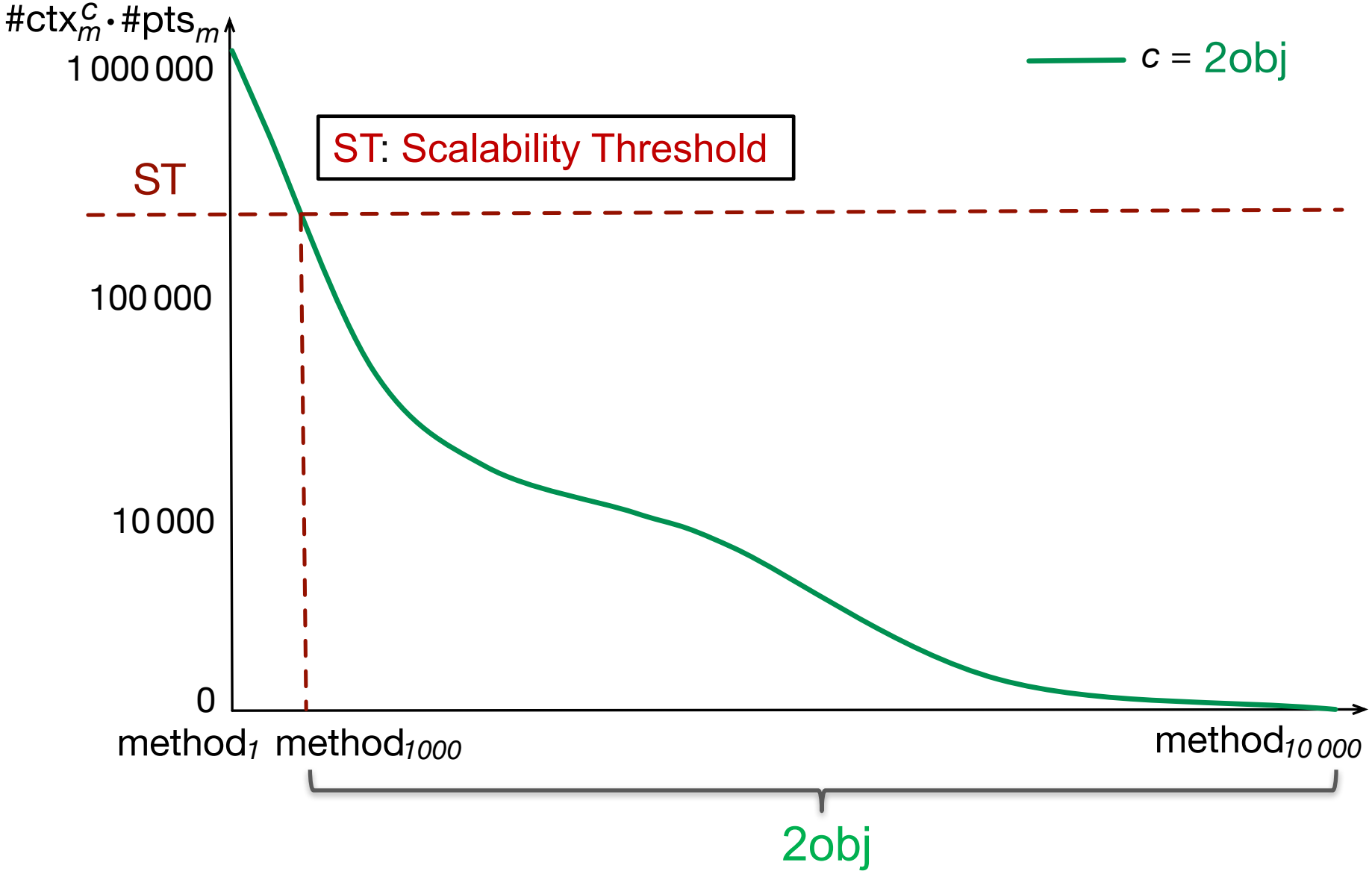
- $\#pts_m$ obtained directly
- $\#ctx_m^c$ obtained by leveraging *Object Allocation Graph** (based on CI)

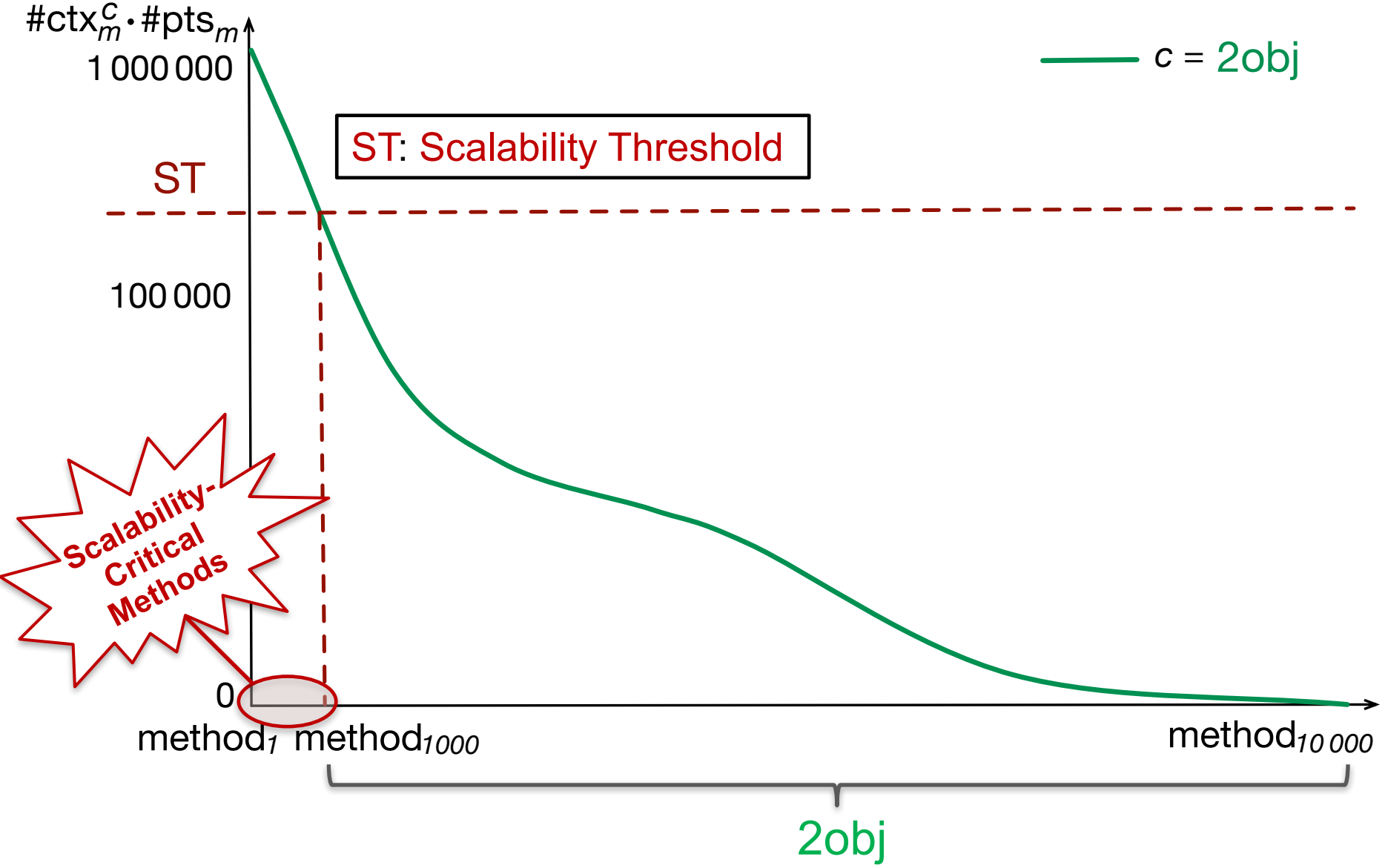
Context estimation problem → Graph traversal problem

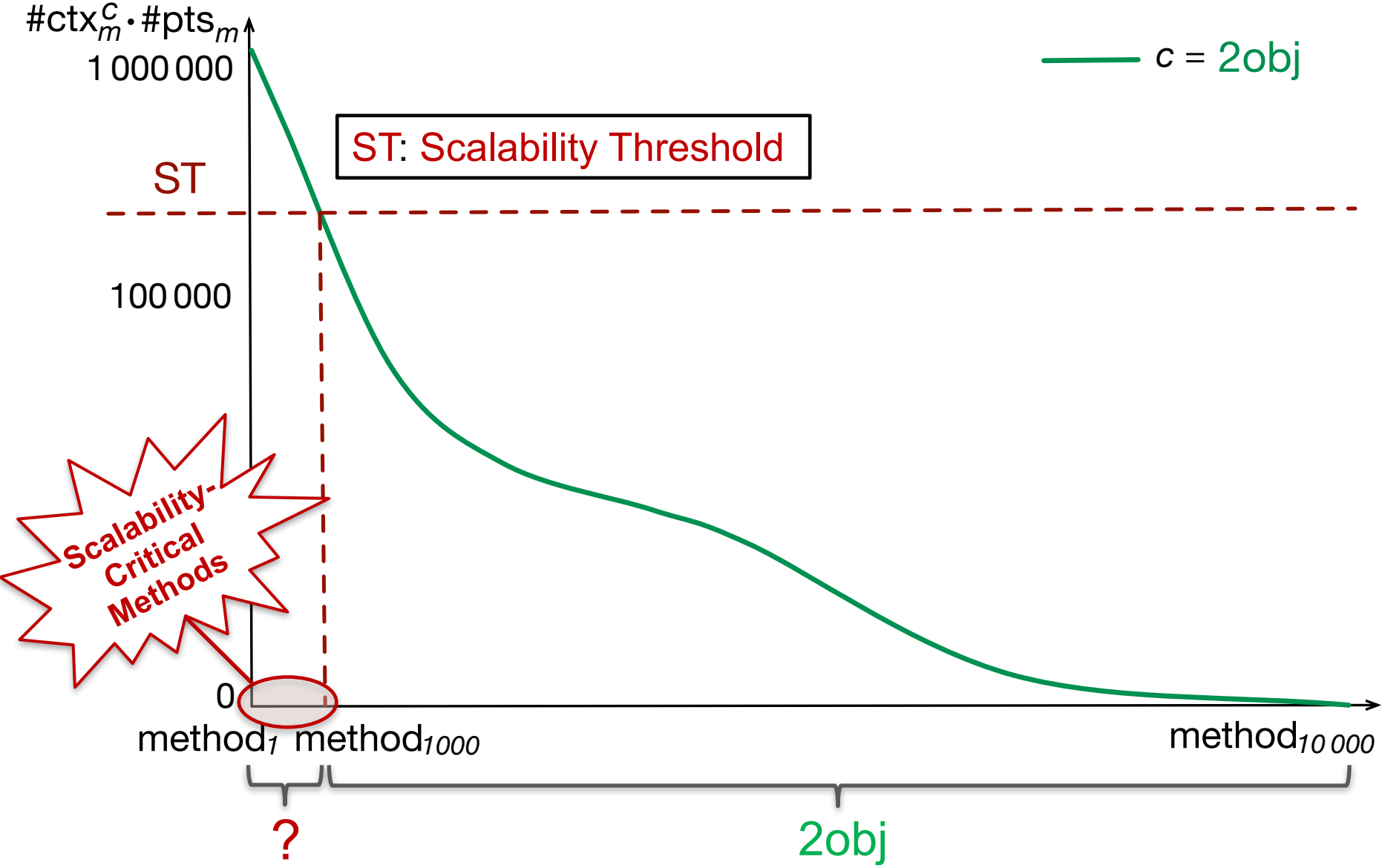
**Making k-object-sensitive pointer analysis more precise with still k-limiting*. *Tan et al. SAS 2016*

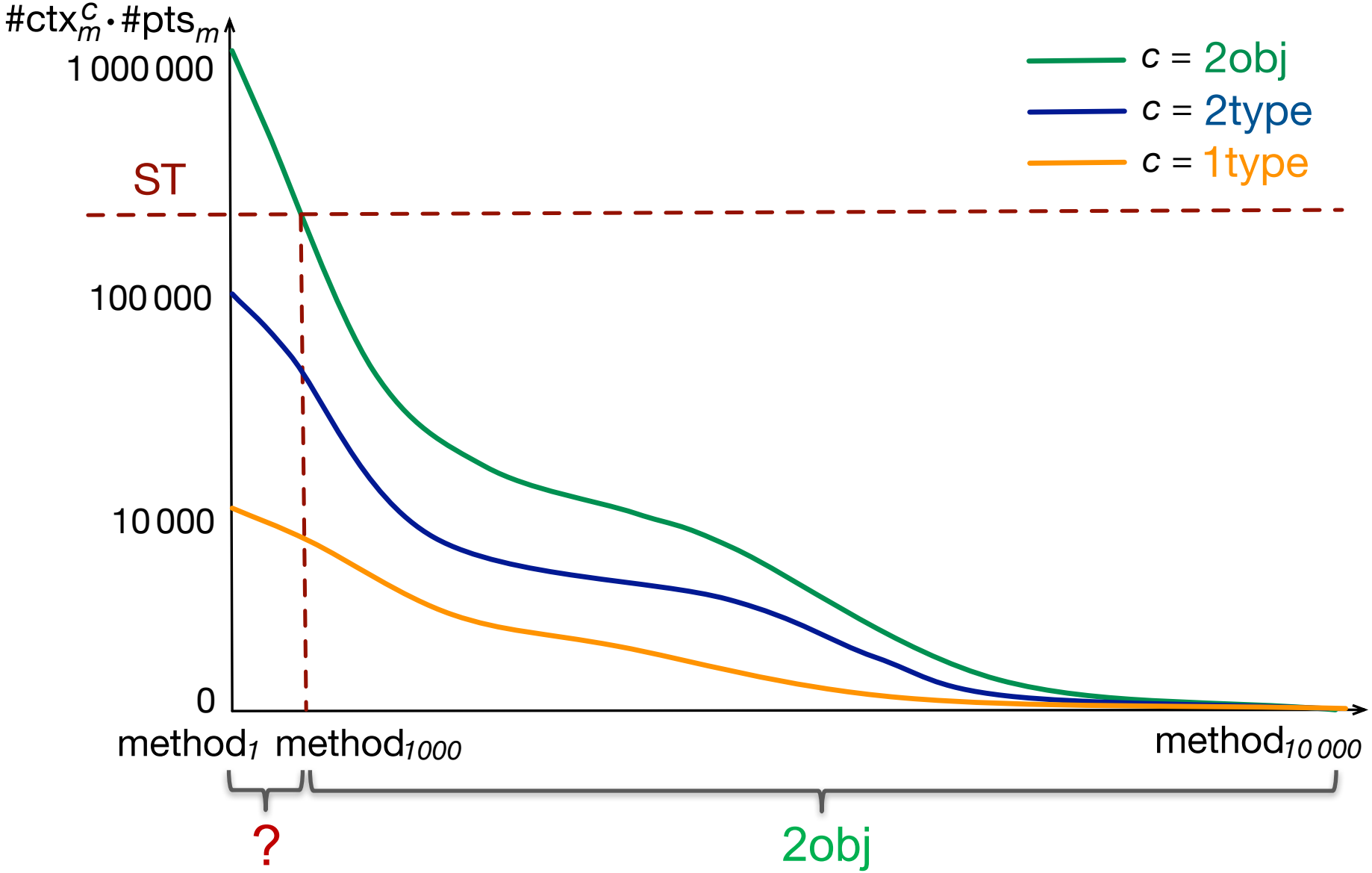
Example

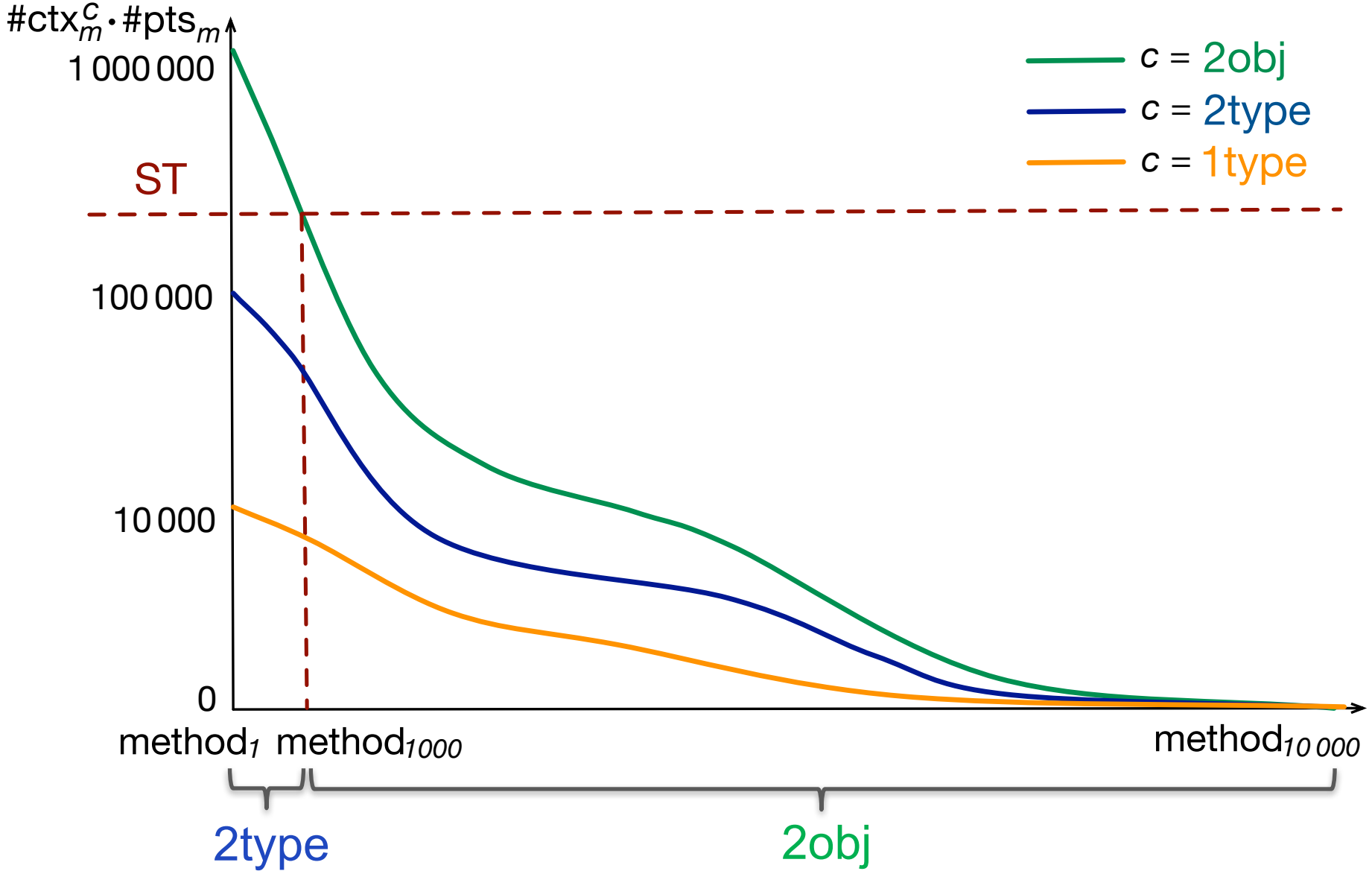

#ctx$_m^c$ · #pts$_m$
1 000 000
100 000
10 000
0
method$_1$
method$_{10\,000}$
c = 2obj

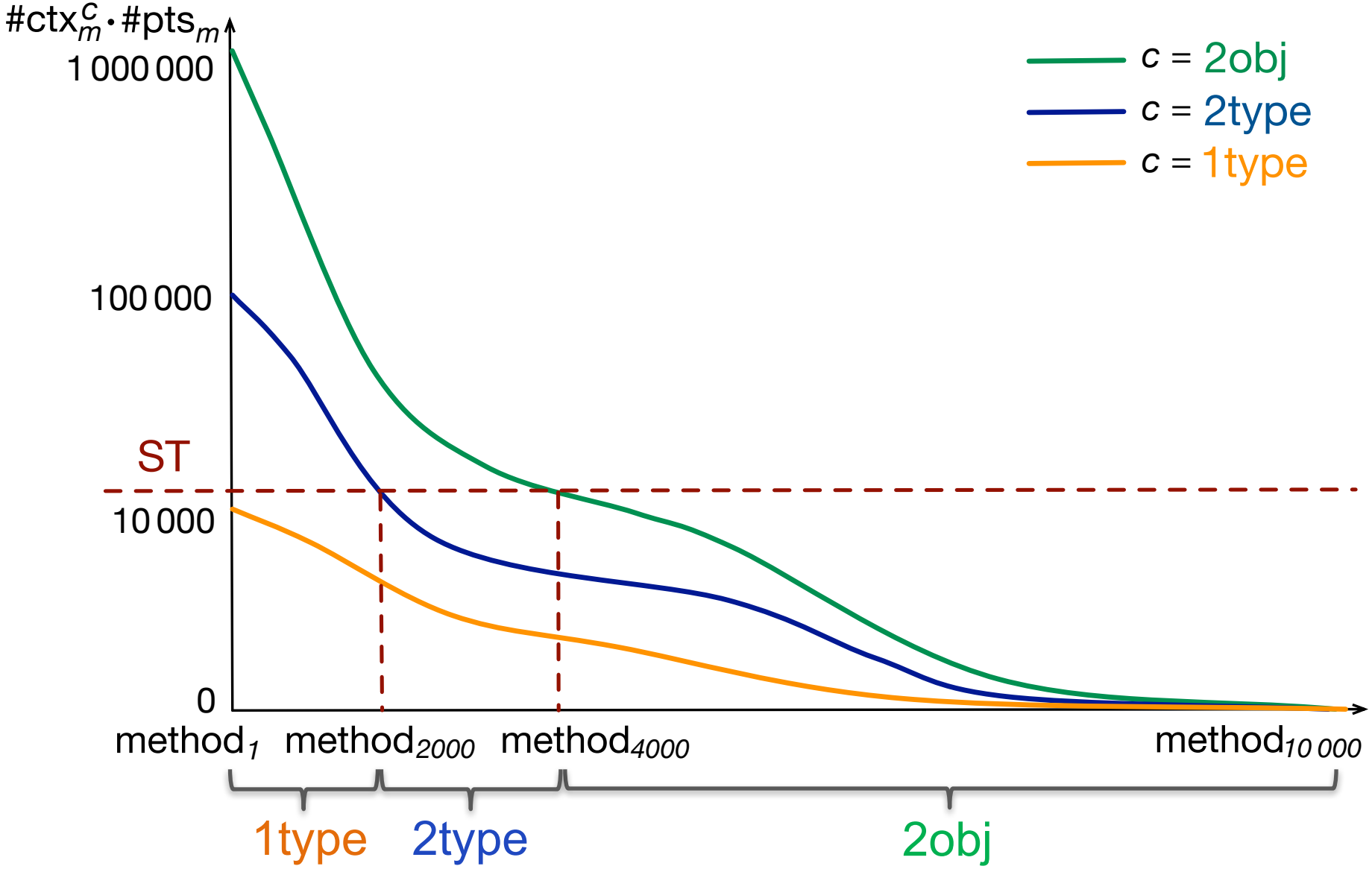

#ctx$_m^c$ · #pts$_m$
1 000 000
100 000
10 000
0
ST
ST: Scalability Threshold
c = 2obj
method$_1$
method$_{1000}$
method$_{10\,000}$
2obj

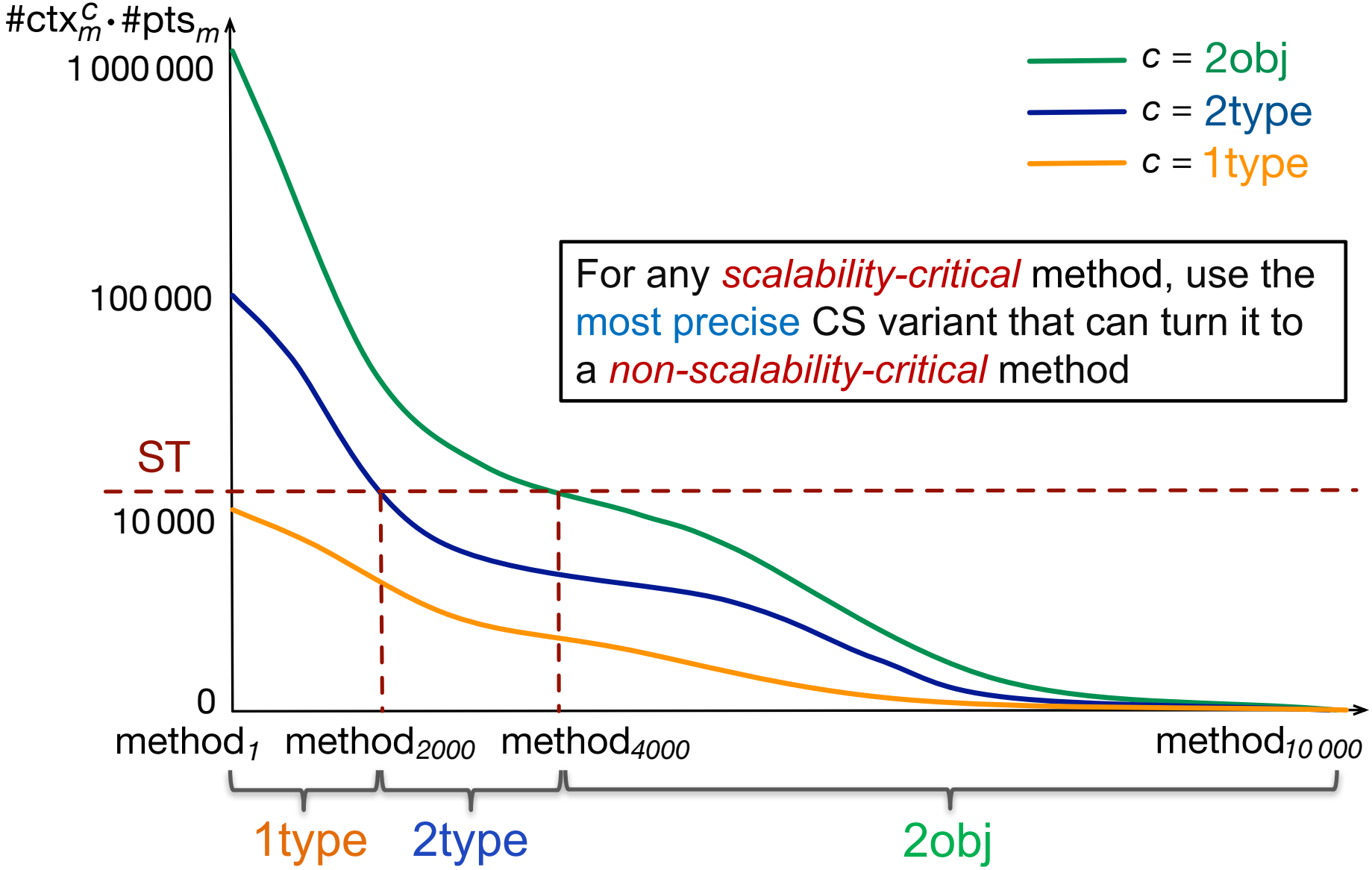

#ctx$_m^c$ · #pts$_m$
1 000 000
100 000
0
ST
ST: Scalability Threshold
c = 2obj
Scalability-Critical Methods
method$_1$
method$_{1000}$
method$_{10\,000}$
2obj

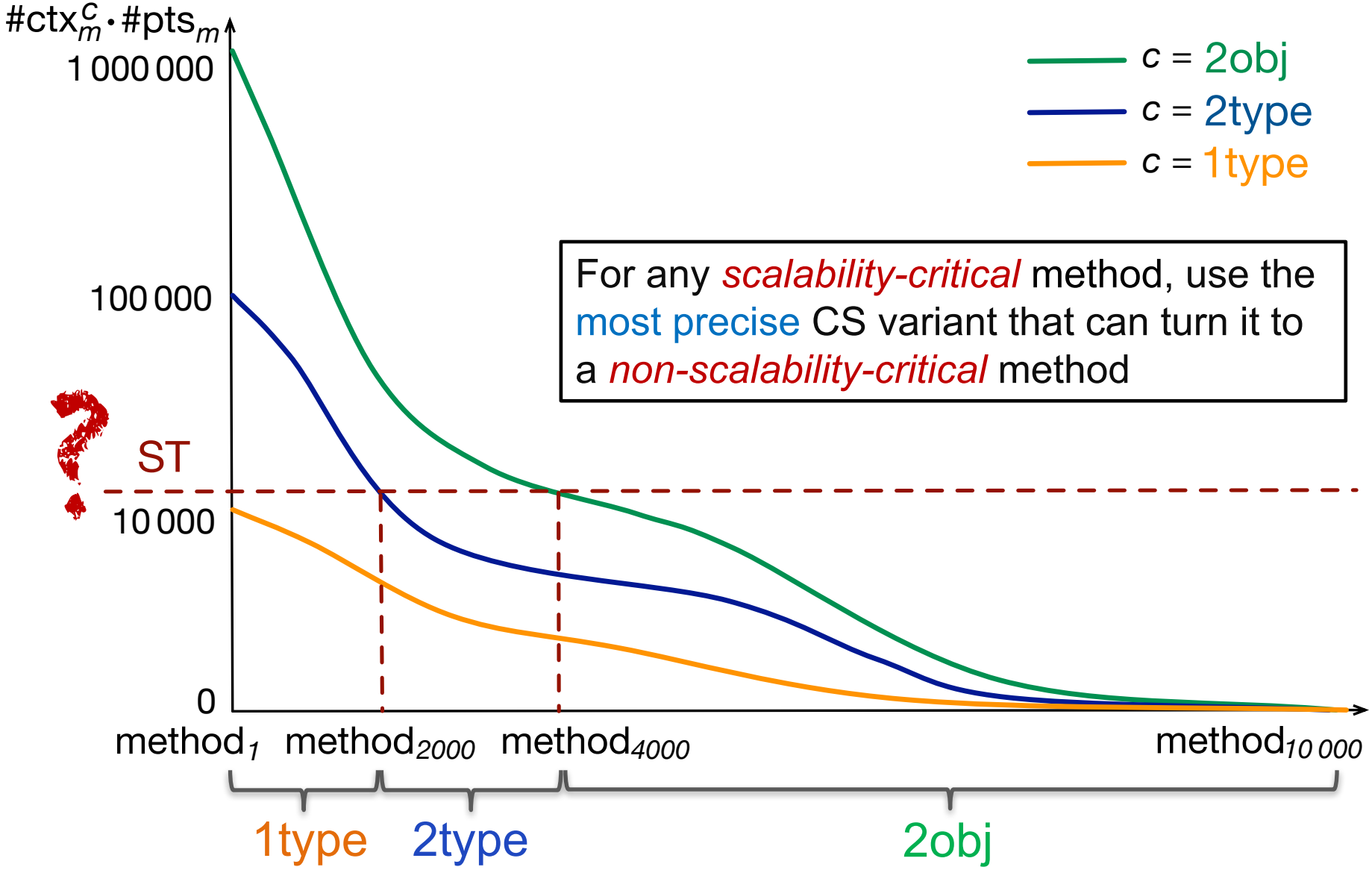

#ctx$_m^c$ · #pts$_m$
1 000 000
100 000
0
c = 2obj
ST: Scalability Threshold
ST
Scalability-Critical Methods
method$_1$
method$_{1000}$
method$_{10\,000}$
?
2obj


#ctx$_m^c$ · #pts$_m$
1 000 000
ST
100 000
10 000
0
c = 2obj
c = 2type
c = 1type
method$_1$
method$_{1000}$
method$_{10\,000}$
?
2obj


#ctx$_m^c$ · #pts$_m$
1 000 000
ST
100 000
10 000
0
c = 2obj
c = 2type
c = 1type
method$_1$
method$_{1000}$
method$_{10\,000}$
2type
2obj


#ctx$_m^c$ · #pts$_m$
1 000 000
100 000
ST
10 000
0
c = 2obj
c = 2type
c = 1type
method$_1$
method$_{2000}$
method$_{4000}$
method$_{10\,000}$
1type
2type
2obj


#ctx$_m^c$ · #pts$_m$
1 000 000
100 000
ST
10 000
0
c = 2obj
c = 2type
c = 1type
For any scalability-critical method, use the most precise CS variant that can turn it to a non-scalability-critical method
method$_1$
method$_{2000}$
method$_{4000}$
method$_{10\,000}$
1type
2type
2obj

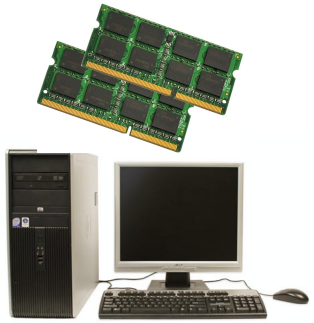

#ctx$_m^c$ · #pts$_m$
1 000 000
100 000
10 000
0
ST
c = 2obj
c = 2type
c = 1type
For any scalability-critical method, use the most precise CS variant that can turn it to a non-scalability-critical method
method$_1$
method$_{2000}$
method$_{4000}$
method$_{10\,000}$
1type
2type
2obj

Total Scalability Threshold (TST)

To automatically choose an appropriate ST_p for different program *p*

Total Scalability Threshold (TST)

To automatically choose an appropriate ST_p for different program *p*

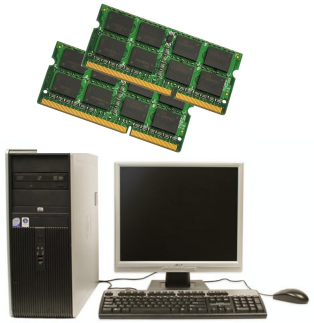


- TST is memory size related
- TST indicates analysis capacity

  How many points-to facts can the memory hold?

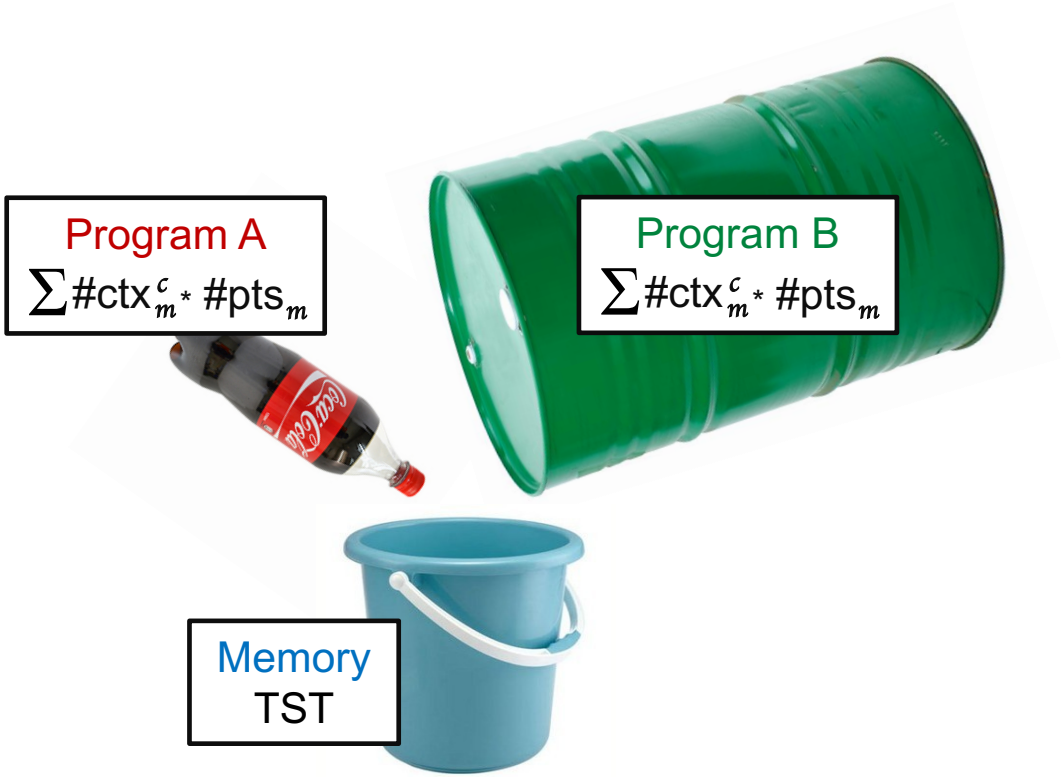
Total Scalability Threshold (TST)

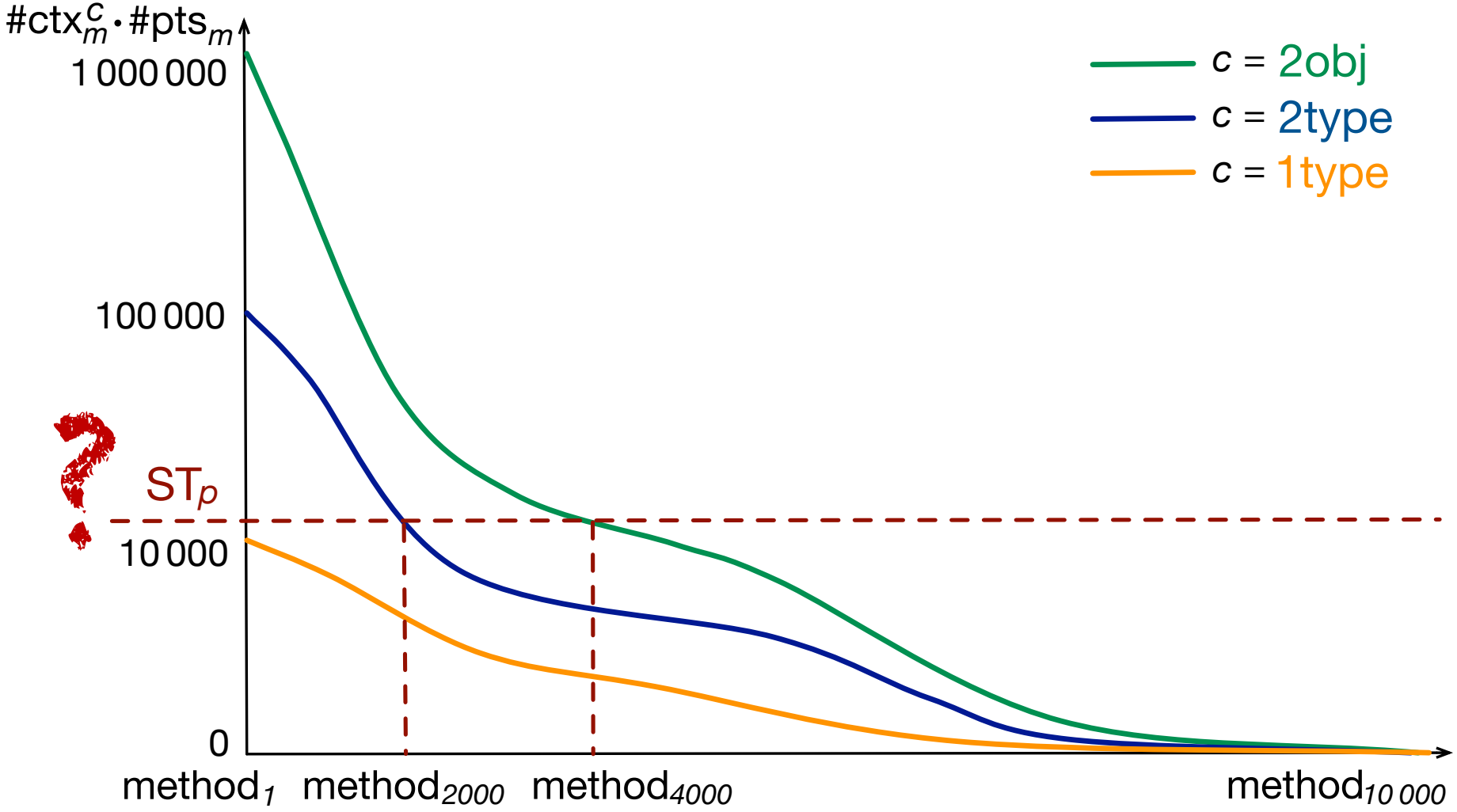
To automatically choose an appropriate ST_p for different program *p*

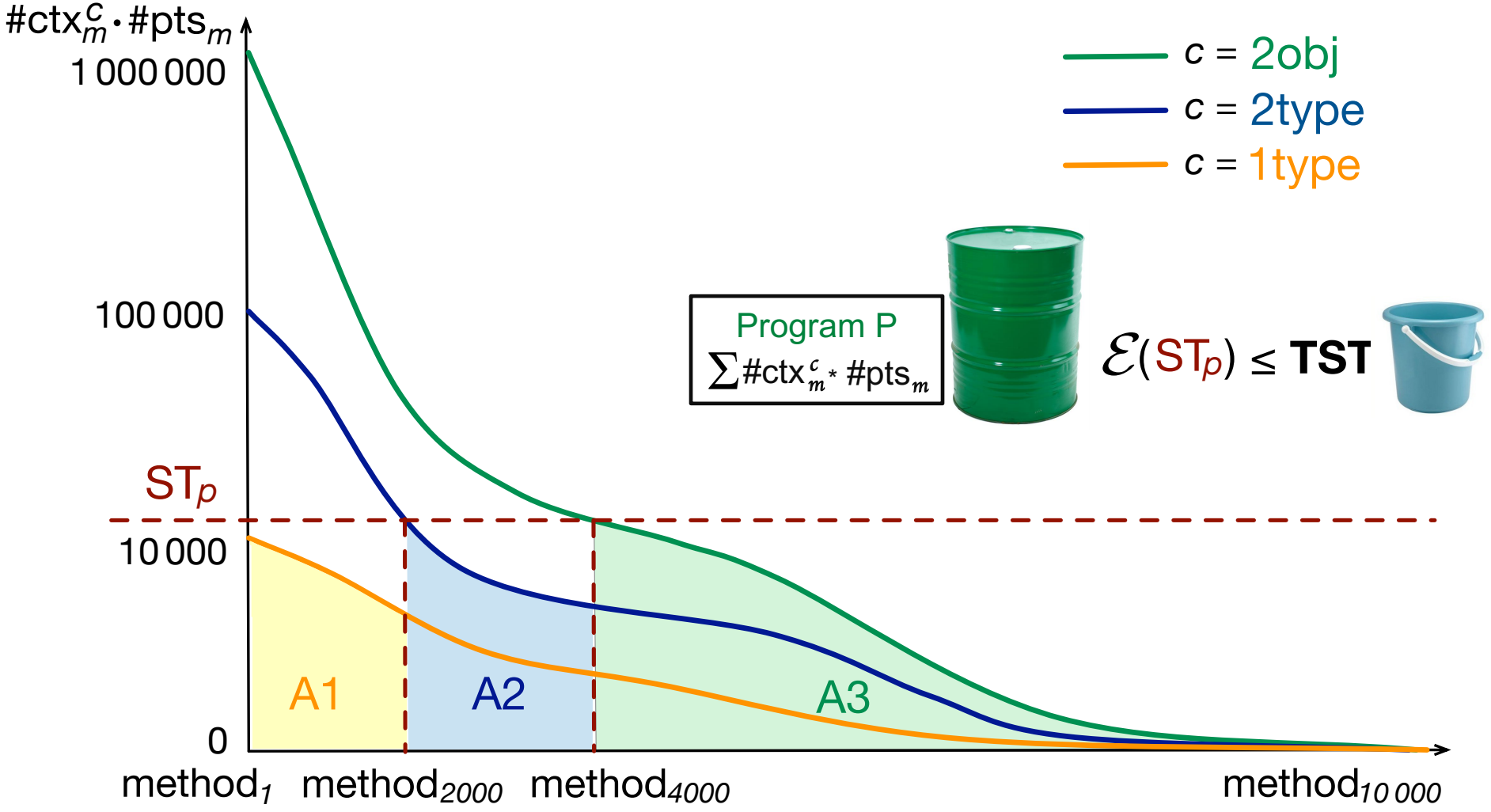


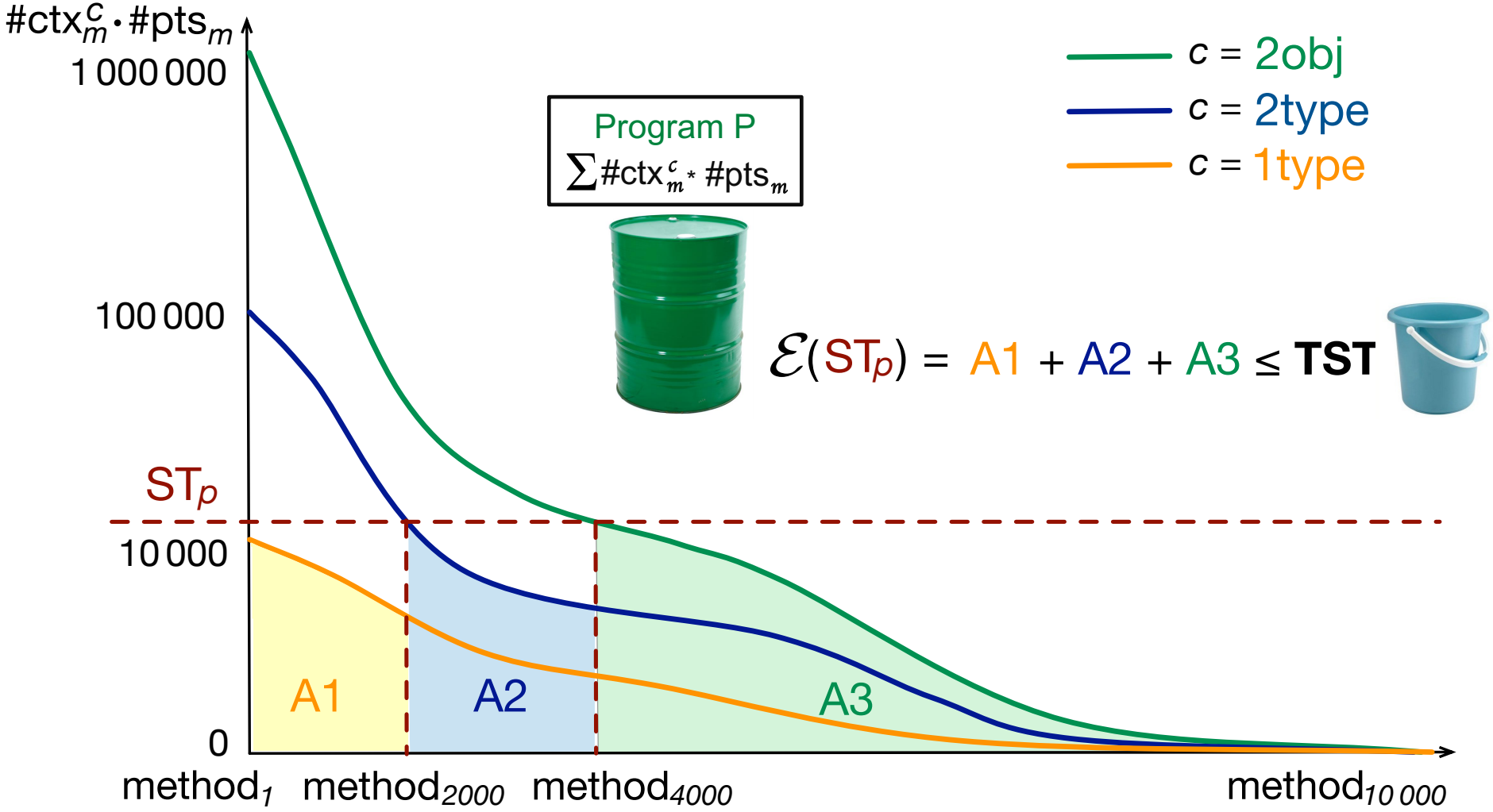
- TST is memory size related
- TST indicates analysis capacity

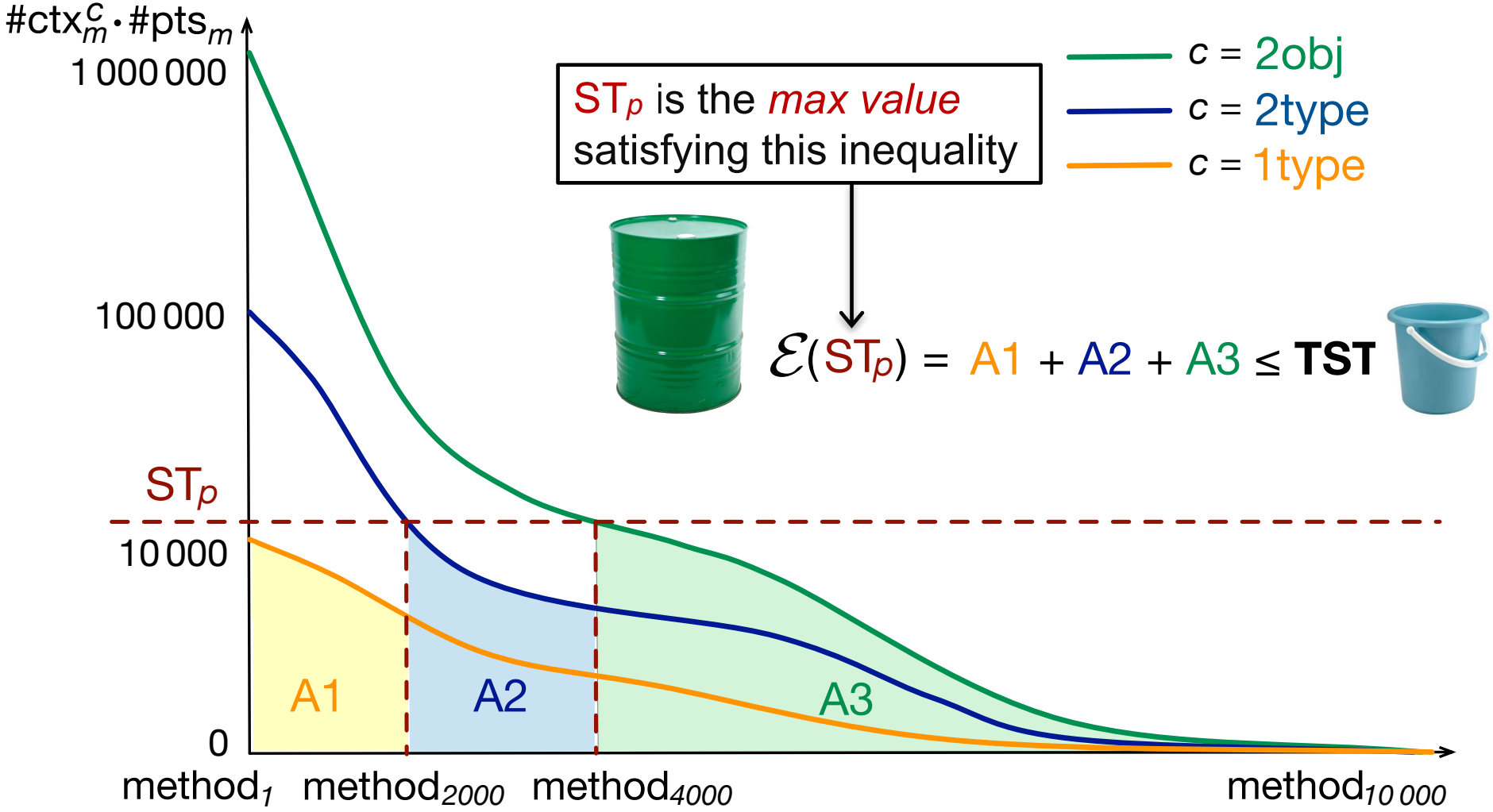
  How many points-to facts can the memory hold?



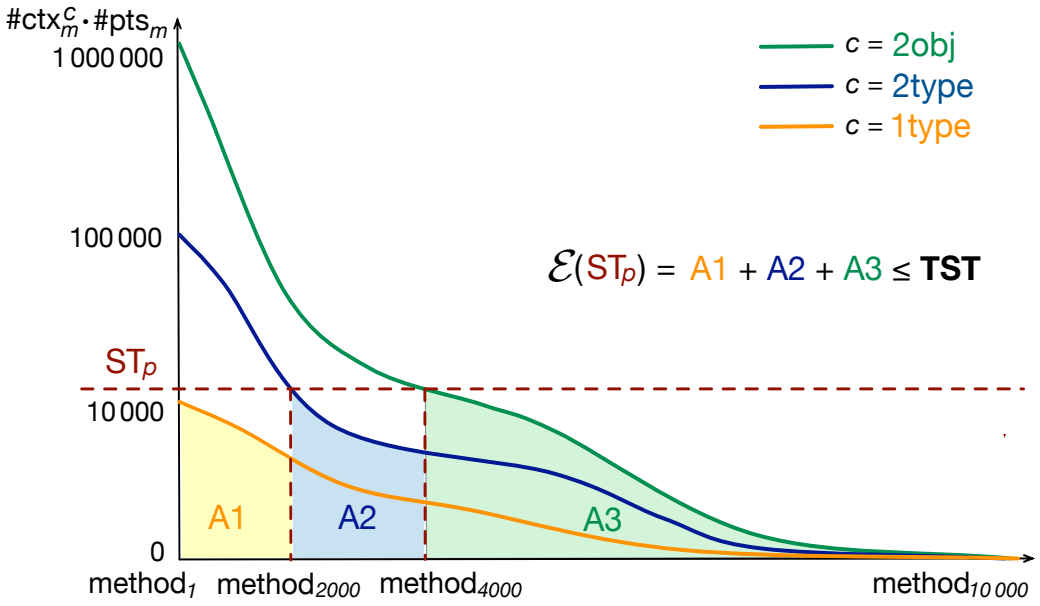

#ctx$_m^c$ · #pts$_m$
1 000 000
100 000
10 000
0
ST$_p$
c = 2obj
c = 2type
c = 1type
method$_1$
method$_{2000}$
method$_{4000}$
method$_{10\,000}$

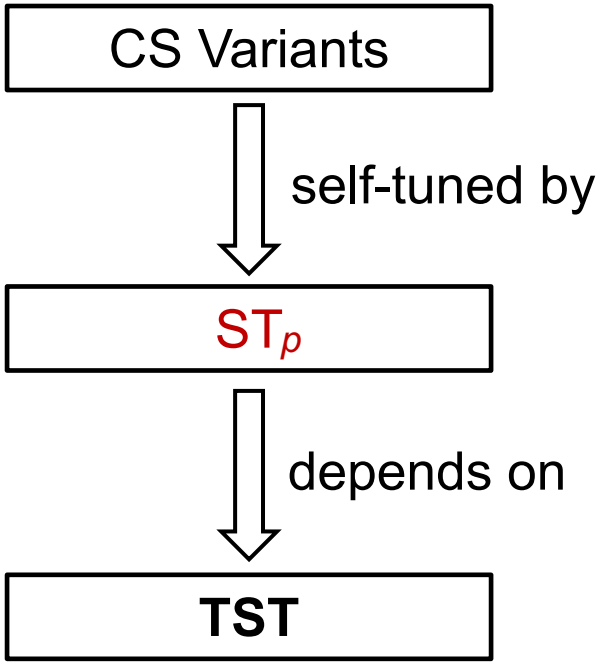

#ctx$_m^c$ · #pts$_m$
1 000 000
100 000
ST$_p$
10 000
0
A1
A2
A3
method$_1$
method$_{2000}$
method$_{4000}$
method$_{10\,000}$
c = 2obj
c = 2type
c = 1type
Program P
$\sum$#ctx$_m^c$ * #pts$_m$
$\mathcal{E}$(ST$_p$) ≤ TST


#ctx$_m^c$ · #pts$_m$
1 000 000
100 000
ST$_p$
10 000
0
A1
A2
A3
method$_1$
method$_{2000}$
method$_{4000}$
method$_{10\,000}$
c = 2obj
c = 2type
c = 1type
Program P
∑#ctx$_m^c$ * #pts$_m$
$\mathcal{E}$(ST$_p$) = A1 + A2 + A3 ≤ TST


#ctx$_m^c$ · #pts$_m$
1 000 000
100 000
ST$_p$
10 000
0
A1
A2
A3
method$_1$
method$_{2000}$
method$_{4000}$
method$_{10\,000}$
ST$_p$ is the *max value* satisfying this inequality
c = 2obj
c = 2type
c = 1type
$\mathcal{E}$(ST$_p$) = A1 + A2 + A3 ≤ **TST**

Scalability-First Pointer Analysis with Self-Tuning Context Sensitivity





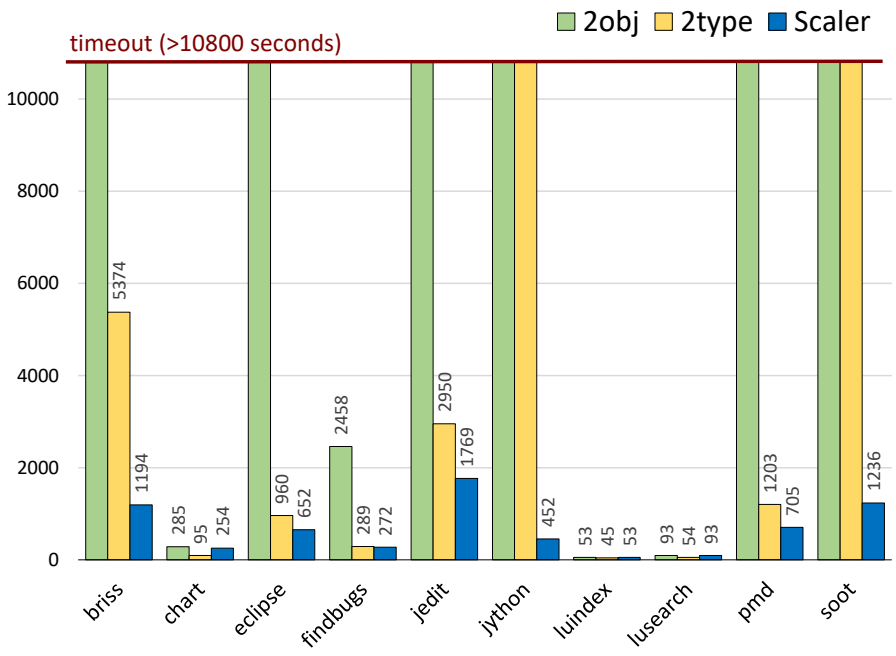
Results

10 Popular Java Programs

Settings

- TST = 30M (48G Memory)
  - *20M, 40M, 60M, etc. are all ok*
  - *Larger TST means better precision but worse efficiency*
- Time budget = 3 hours (per program)

Results



Scaler

Scalability

as good as CI

Precision

comparable to or *better than* the best scalable CS

Settings

- TST = 30M (48G Memory)
  - *20M, 40M, 60M, etc. are all ok*
  - *Larger TST means better precision but worse efficiency*
- Time budget = 3 hours (per program)

Results

Complex program

Medium-Complexity program

Simple program

Scalability

as good as CI

Precision

comparable to or *better than* the best scalable CS

Complex program

Analysis	Time (seconds) 3h = 10800s	Precision Metrics			
		#may-fail casts	#poly calls	#reachable methods	#call graph edges
CI	112	2234	2778	12718	114856
2obj → 2type → 1type	**>3h** + **>3h** + 1997	2117	2577	12430	111834
Scaler	452	1852	2500	12167	107410

In all cases, *lower is better*

Medium-Complexity program

Analysis	Time (seconds) 3h = 10800s	Precision Metrics			
		#may-fail casts	#poly calls	#reachable methods	#call graph edges
CI	49	2508	2925	13036	77370
2obj ~~→ 2type → 1type~~	2458	1409	2182	12657	65836
Scaler	272	1452	2195	12676	66177

In all cases, *lower is better*

Simple program

Analysis	Time (seconds) 3h = 10800s	Precision Metrics			
		#may-fail casts	#poly calls	#reachable methods	#call graph edges
CI	22	734	940	6670	33130
2obj ~~→ 2type → 1type~~	53	297	675	6256	29021
Scaler	53	297	675	6256	29021

In all cases, *lower is better*

Analysis	Time (seconds) 3h = 10800s	Precision Metrics			
		#may-fail casts	#poly calls	#reachable methods	#call graph edges
CI	112	2234	2778	12718	114856
2obj → 2type → 1type	>3h + >3h + 1997	2117	2577	12430	111834
Scaler	452	1852	2500	12167	107410

Analysis	Time (seconds) 3h = 10800s	Precision Metrics			
		#may-fail casts	#poly calls	#reachable methods	#call graph edges
CI	49	2508	2925	13036	77370
2obj ~~→ 2type → 1type~~	2458	1409	2182	12657	65836
Scaler	272	1452	2195	12676	66177

Analysis	Time (seconds) 3h = 10800s	Precision Metrics			
		#may-fail casts	#poly calls	#reachable methods	#call graph edges
CI	22	734	940	6670	33130
2obj ~~→ 2type → 1type~~	53	297	675	6256	29021
Scaler	53	297	675	6256	29021

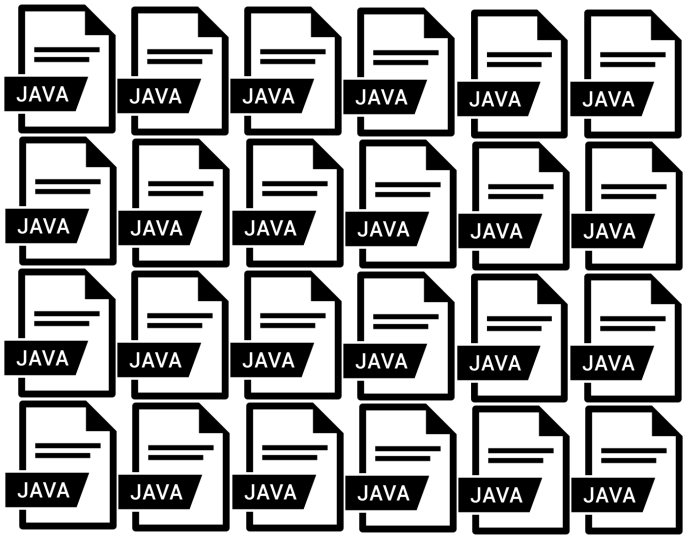
Analysis	Time (seconds) 3h = 10800s	Precision Metrics			
		#may-fail casts	#poly calls	#reachable methods	#call graph edges
CI	112	2234	2778	12718	114856
2obj → 2type → 1type	>3h + >3h + 1997	2117	2577	12430	111834
Scaler	452	1852	2500	12167	107410

Analysis	Time (seconds) 3h = 10800s	Precision Metrics			
		#may-fail	#poly	#reachable	#call graph edges
CI					77370
2obj → 2type → 1type					65836
Scaler	272	1452	2195	12676	66177

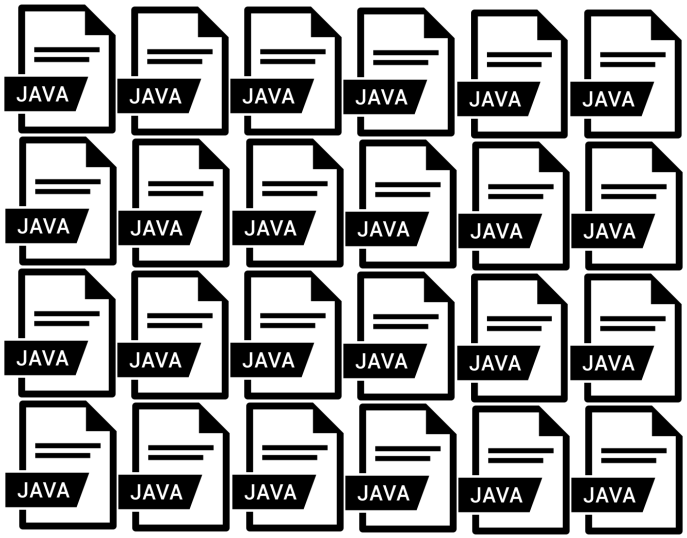
Scaler: Good Scalability & High Precision
regardless of the program being analyzed

Analysis	Time (seconds) 3h = 10800s	Precision Metrics			
		#may-fail casts	#poly calls	#reachable methods	#call graph edges
CI	22	734	940	6670	33130
2obj → 2type → 1type	53	297	675	6256	29021
Scaler	53	297	675	6256	29021

Want to have a good night's sleep?



Want to have a good night's sleep?



Scaler

Good Scalability & High Precision

Conclusion

- **Artifact** successfully evaluated
- **Open Source Tool** available

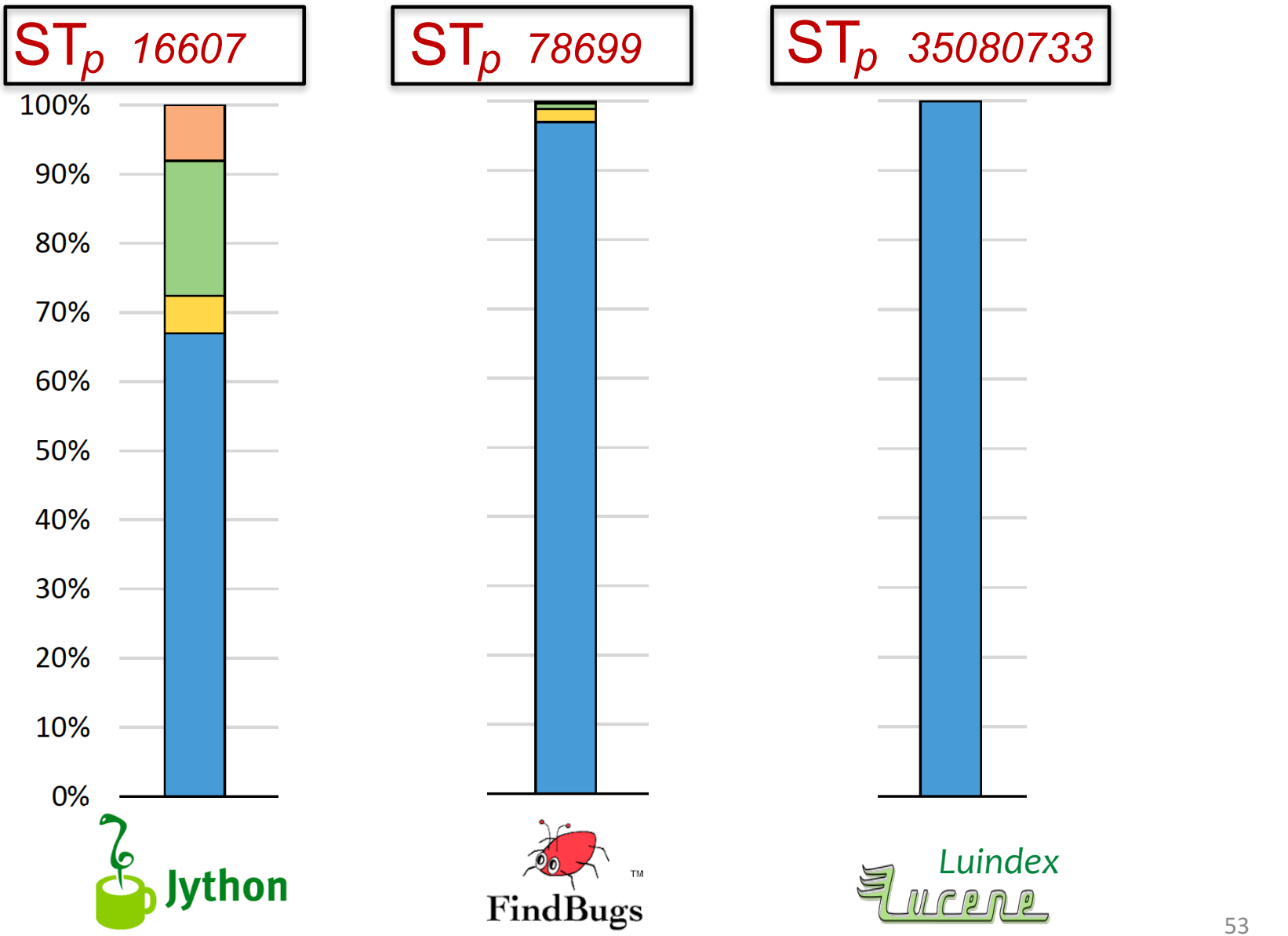
http://www.brics.dk/scaler

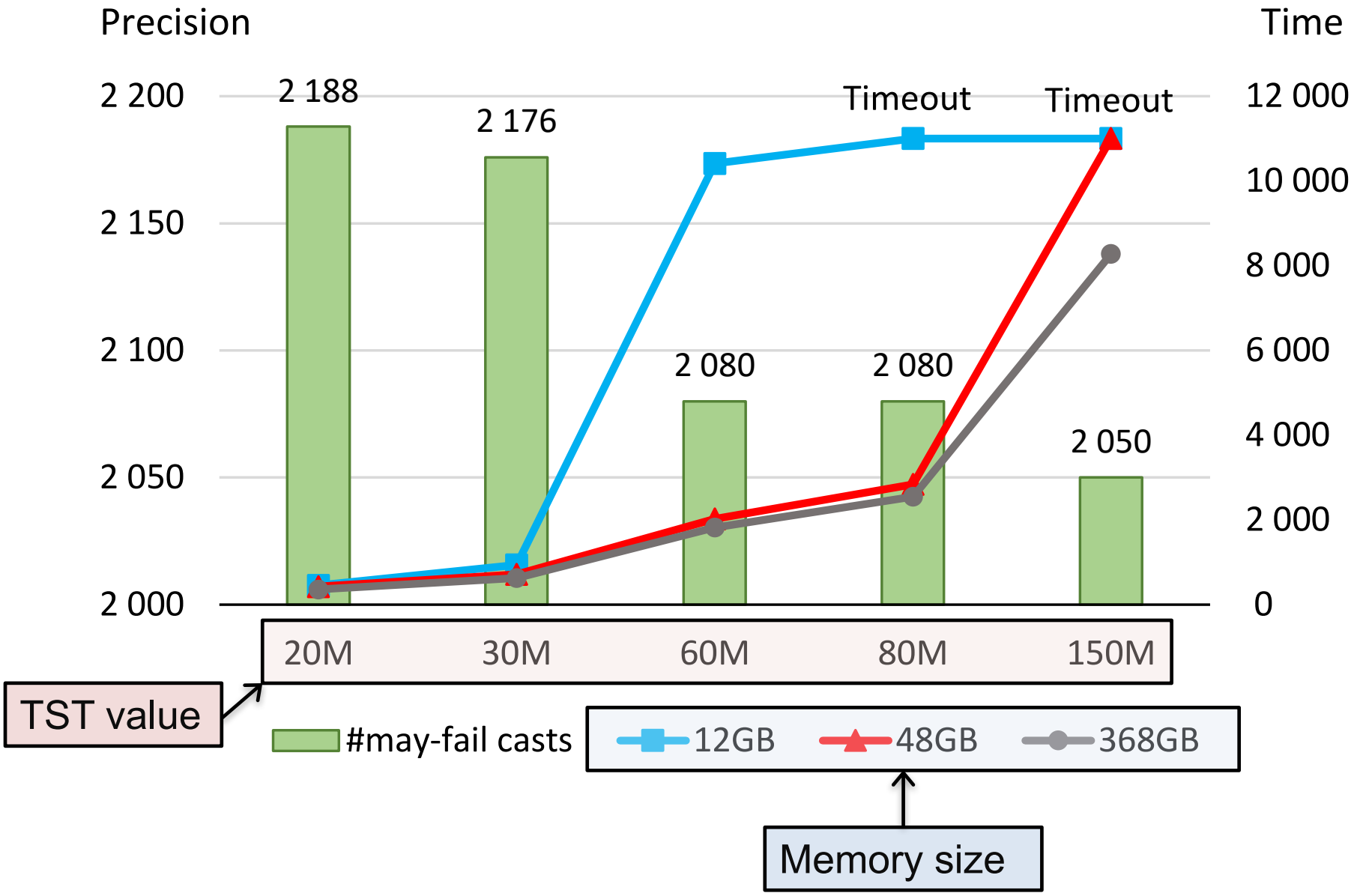
- Predict #CS-points-to facts using fast pre-analysis
- Memory-related scalability (TST)
- Method-level CS configurations (ST)

- Extremely good scalability + high precision (one shot)
- Directly benefit many software engineeing tasks/tools

- Expect the idea behind help scale other static analyses

$\mathcal{E}(ST_p) \leq$ **TST** (30M)




Precision
Time
2 200
2 150
2 100
2 050
2 000
12 000
10 000
8 000
6 000
4 000
2 000
0
2 188
2 176
2 080
2 080
2 050
Timeout
Timeout
20M
30M
60M
80M
150M
TST value
#may-fail casts
12GB
48GB
368GB
Memory size